Lista 2 – LAEDs II

Nome: Gustavo de Assis Xavier

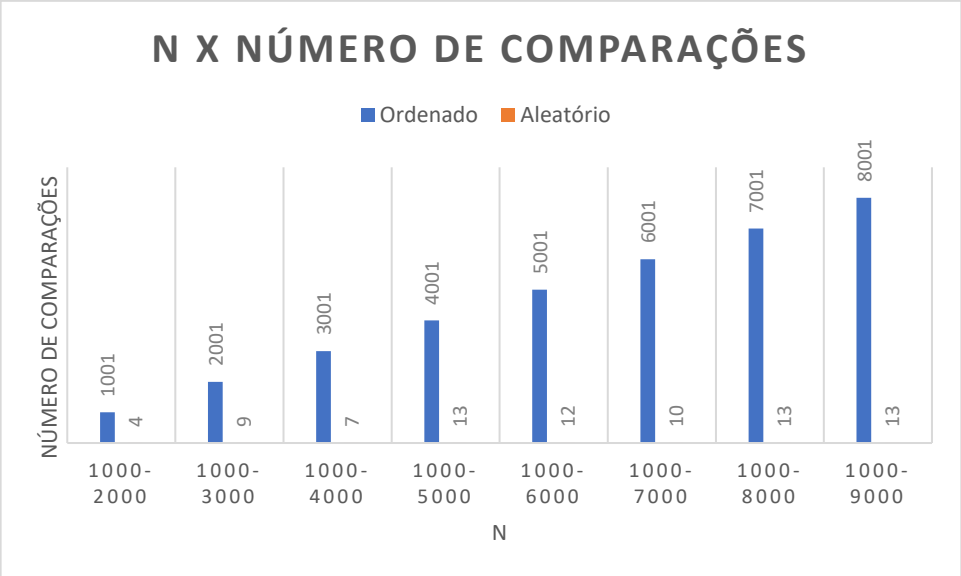
Esse trabalho consiste na análise dos gráficos de dois testes que se baseiam em gerar e pesquisar em árvores binárias não balanceadas, com elementos ordenados e aleatórios.

Os testes consistes em criar diversas árvores com N elementos que variam de 1000 a 9000, divididos em intervalos de 1000, ou seja, a primeira árvore contém elementos de 1000 a 2000, a segunda árvore possuí elementos de 1000 a 3000, e assim por diante.

Os resultados obtidos dos testes são apresentados pela tabela abaixo:

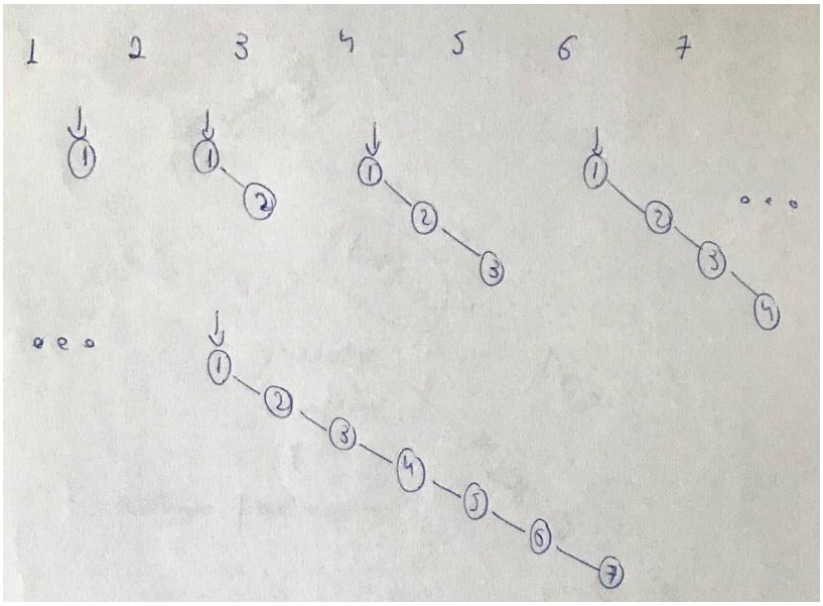
	Sem balanceamento			
	Ordenada		Aleatória	
N	Comparações	Tempo	Comparações	Tempo
1000-2000	1001	2813180	4	112690
1000-3000	2001	1178230	9	52110
1000-4000	3001	1204180	7	57690
1000-5000	4001	276150	13	52760
1000-6000	5001	373220	12	60970
1000-7000	6001	424530	10	64960
1000-8000	7001	610490	13	79490
1000-9000	8001	683860	13	303530

Extraindo da tabela e representando por gráficos, o número de comparações fica:



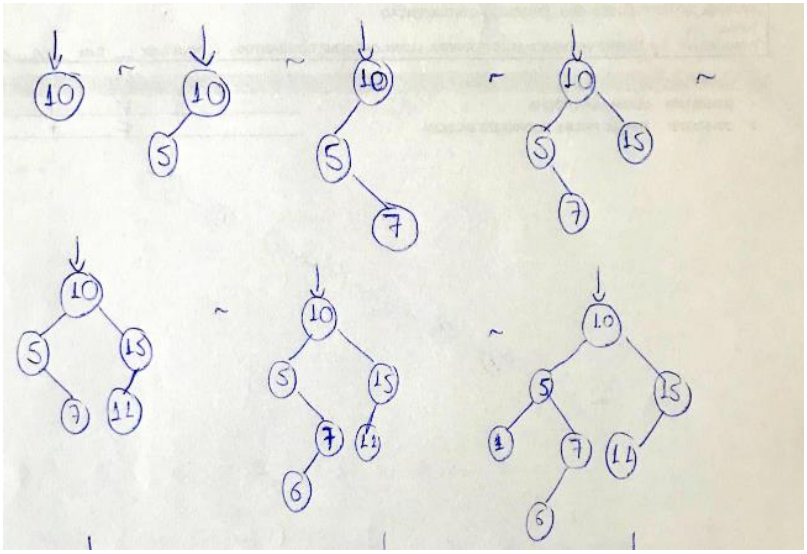
É possível notar uma diferença exorbitante nas duas abordagens. Essa diferença se deve ao fato de que o pior caso da árvore binária não balanceada é quando as chaves são inseridas na ordem crescente ou decrescente, nesses casos, a árvore obtida é uma lista linear. Então para achar um elemento que não exista na lista (considerando que ele esteja depois de qualquer elemento da árvore), seria necessário comparar com todos os elementos nela presente. Nesse tipo de árvore, o número médio de comparações é (n+1)/2 e se trata de um método o(n).

Exemplo de uma árvore não balanceada com números inseridos na ordem crescente:

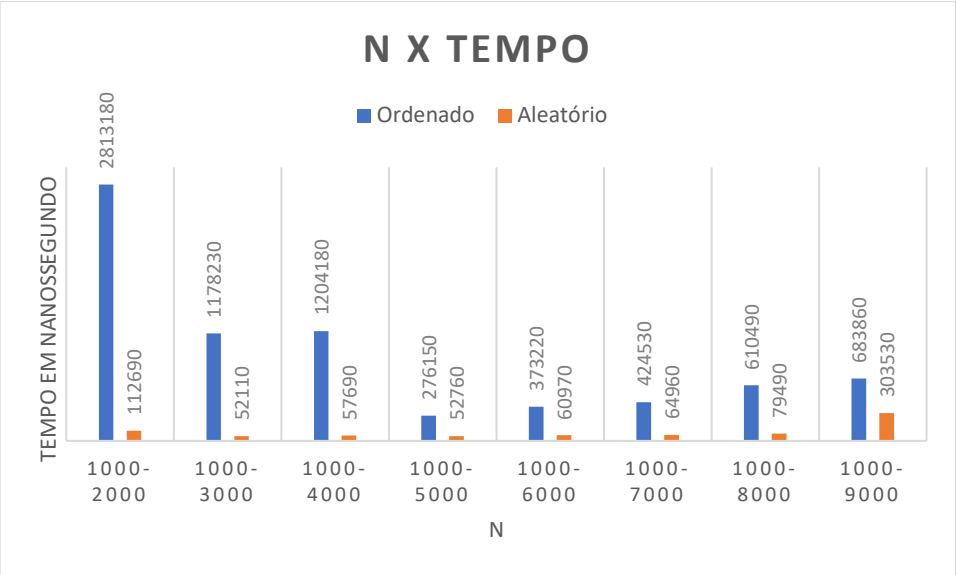


Já quando temos os números aleatórios sendo inseridos na árvore, temos que ela cresce de forma mais ramificada, permitindo assim, que o número de comparações seja extremamente reduzido. Em todos os testes feitos por mim dessas duas árvores, na inserção aleatória, obtive resultados entre 4 e 20 comparações na pesquisa de um número inexistente. Nesse tipo de caso, o número médio de comparações para recuperar um registro é 1,39 log n.

Exemplo de uma árvore não balanceada com números inseridos em ordem aleatória:



Já comparando o tempo temos:



Como esperado, devido ao número muito reduzido de comparações, o tempo gasto na pesquisa em uma árvore com números aleatórios é muito menor. No caso apresentado, o tempo é 10 vezes menor. Lembrando que o tempo não é uma medida tão eficaz, não nesse caso, devido às múltiplas variáveis que afetam a medida do tempo, como programas em segundo plano e/ou otimizações que o compilador faz no código. Mas em todo caso, é possível notar claramente a superioridade da árvore binária de números aleatórios.